8th NASCUG Meeting

CORBA Based Co-Verification Methodology For SystemC

by Pascal Giard
Jean-François Boland, Jean Belzile
M.Ing. Student
École de technologie supérieure
Motivation

OPERATION Project
- Array of homogeneous processing elements
- Models in MATLAB, SystemC, C++, VHDL, etc.
- Needs a backplane!

C++ Model

Any Model

Simulation Backplane

OSCI Simulator

Scheduler

Ext. IF

C

C

C

C

C

MEM

MATLAB/Simulink

C++ GUI

GHDL/ModelSim

FPGA Board
Problem description

- Design complexity
- Various modeling languages: heterogeneous design and verification components
- Specialized and complementary simulators and/or tools excelling in their respective domains
- Design architecture may change
Outline

- Proposed Methodology
  - Communication
  - Data Adaptation
  - Tools Involved

- Results
  - Demo
  - Observed Benefits

- Conclusion & Future Work
Proposed Methodology (1/2)

- Communication
  - Make simulators and tools communicate in a standard way
  - Use CORBA as a simulation backplane
Proposed Methodology (2/2)

- Data adaptation
  - Convert data where abstraction levels have different representations
  - Implement transactor concept

![Diagram showing component adaptation and signaling via CORBA Backplane]
Communication: Why CORBA? (1/2)

- Broadly accepted and used architecture
  - From military to free/libre software hobbyists
  - From real-time hardware to software

- Language and platform independent
  - Code generators for a plethora of languages e.g. C/C++, Ada, Python and VHDL
  - Implementation for various platforms such as MS Windows, GNU/Linux, MacOS X, etc.
Communication: Why CORBA? (2/2)

- Supports various transport and pseudo-transport protocols
  - e.g. TCP/IP, SSH, shared memory and file.
- Extensible Transport Framework allows addition of unsupported protocols
- Hardware support starts to appear
  - PrismTech’s Integrated Circuit ORB
  - Objective Interface Systems’ ORBexpress FPGA
- Allows distributed processing
Communication: CORBA Introduction

- Component interfaces are described using the Interface Definition Language (IDL)
- Any tool/simulator can be used as a master or slave as long as it can integrate an ORB server or client
- A Naming Service keeps track of the ORBs providing objects (components)
- Any transport protocol can be used as long as the Naming Service supports it (ETF allows more)
Communication: Layered Structure

**Master Simulator /Tool**
- Component Proxy
  - CORBA Component Wrapper
  - CORBA Client Wrapper
  - CORBA Client

**Simulator /Tool**
- Component
  - CORBA Component Wrapper
  - CORBA Server Wrapper
  - CORBA Server

**CORBA Component Wrapper**
Transactor, performs signal adaptation and synchronization

**CORBA Client /Server Wrapper**
Handles CORBA Client /Server initialization and destruction

**CORBA Client /Server**
Autogenerated Client /Server requesting/providing components

**CORBA Naming Service**
Makes the connection between components requesters/providers.

**Link**
Implements any transport protocol supported by the CORBA Naming Service e.g. TCP/IP
Communication: Integrating CORBA to SystemC

- **Required steps:**
  - Defining components interfaces (IDL file)
  - Autogenerate CORBA servers and clients (feeding IDL file to code generator)
  - Write CORBA Wrappers for...
    - Components
    - Servers
    - Clients

- Easy to implement once we’ve been through the first wrapper
- Less than week to implement demo
Data Adaptation: Transactor examples

- Component Wrappers also are transactors
- Transactors can either be on the client or server side
Tools Involved

- MathWorks MATLAB
  - Algorithmic Modeling
- Open SystemC Initiative Simulator
  - System Level Design
- GHDL
  - RTL Modeling
- TAO
  - CORBA Backplane
  - Code generation
- GCC and Autotools
  - Configuration and compilation
Results/Demonstration (1/2)

- Design a FIR filter from algorithmic modeling to RTL design
- One testbench
- Generate data with MATLAB/Simulink
- Feed data to all implementations:
  - Algorithmic level (MATLAB)
  - System level – UTF (SystemC)
  - System level – TF (SystemC)
  - RTL (VHDL)
- Display/validate results in MATLAB/Simulink
Results/Demonstration (2/2)
Results: Observed Benefits

- Seamless integration of multiple:
  - Levels of abstraction
  - Modeling languages
  - “Physical” location and/or simulators/tools
- Same testbench across all implementations
- Language independence: from Algorithmic level to RTL
- Platform independence: anything that can implement CORBA
- Components can be refined independently
Conclusion

- Simulator/Tool-independent solution addressing heterogeneity
- Integrates to OSCI simulator w/o any modification to the core
- CORBA client/server wrappers hide communication complexity
- Data can be adapted by CORBA Component Wrappers acting as transactors
- Has a distributed topology
- Promotes open standards
Future Work

- Cleaner SystemC integration (TLM-2)
- Performance evaluation/tweaking
- Add support for ModelSim
- Hardware-In-the-Loop with PrismTech ICO
References

- **The ACE ORB**
  - [http://www.cs.wustl.edu/~schmidt/TAO.html](http://www.cs.wustl.edu/~schmidt/TAO.html)

- **SystemC**
  - [http://www.systemc.org](http://www.systemc.org)

- **MathWorks MATLAB**
  - [http://www.mathworks.com](http://www.mathworks.com)

- **GHDL**
  - [http://ghdl.free.fr](http://ghdl.free.fr)

- **PrismTech ICO**
  - [http://www.prismtech.com](http://www.prismtech.com)
Questions?

Thank you for listening!

Contact me at

Pascal.Giard@lacime.etsmtl.ca